Visualizing Social Media Content with SentenTree

Mengdie Hu, Krist Wongsuphasawat, and John Stasko

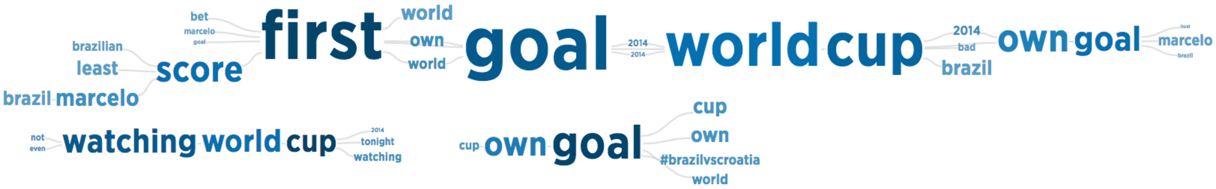


Fig. 1: Part of a SentenTree visualization of a collection of 189,450 tweets (108,702 unique) posted in a 15 minute time window around the first goal of the opening game of the 2014 Soccer World Cup.

Abstract—We introduce SentenTree, a novel technique for visualizing the content of unstructured social media text. SentenTree displays frequent sentence patterns abstracted from a corpus of social media posts. The technique employs design ideas from word clouds and the Word Tree, but overcomes a number of limitations of both those visualizations. SentenTree displays a node-link diagram where nodes are words and links indicate word co-occurrence within the same sentence. The spatial arrangement of nodes gives cues to the syntactic ordering of words while the size of nodes gives cues to their frequency of occurrence. SentenTree can help people gain a rapid understanding of key concepts and opinions in a large social media text collection. It is implemented as a lightweight application that runs in the browser.

Index Terms—text visualization, social media, natural language processing, word cloud, Twitter

1 Introduction

With the popularity of social media and online communities, a new type of text document is growing explosively. Examples of these documents include Tweets, Facebook posts, YouTube comments, and Yelp reviews that we will collectively call *social media text*. Social media text encodes rich information on the public's interests and opinions and this information is valuable to both professional analysts and casual users. While we can gain valuable information from examining the social network structure and the message volume fluctuations, making sense of the textual content itself remains very challenging. Due to the natural complexities of human languages, unstructured text does not lend itself well to computational analysis. Social media text further complicates the problem with its unique qualities—Compared to traditional text documents (e.g. book chapters, news articles) a social media text collection typically contains a large number of very short documents authored by different users. For a given topic, we can accumulate a huge document collection in a very short period of time that contains highly repetitive and redundant information. Keeping these challenges in mind, we focus on a simple goal in this paper: designing a visualization technique that helps people gain a quick overview of the content of a social media text collection.

There are typically two ways to provide a high-level summary of a text document set. One solution is to extract a few representative sentences from the collection. For social media messages, this can be accomplished by showing the documents with the most shares or "favorite" designations. This popularity-based solution works well for some situations, but it runs a risk of lacking coverage. Research shows that the most popular messages on social media are usually produced by a small population of elite users or opinion leaders [19,42]. Therefore, selecting the most popular messages usually means overlooking the voices of "ordinary" users of social media. In many scenarios, the opinions of ordinary people are precisely what the analyst wants to hear.

A second solution that takes coverage into consideration is to extract some common information from the entire document collection. Numerous research efforts in the text mining community employ advanced rule-based and statistical methods (e.g., entity identification, topic modeling) to produce representative word lists or distributions and document clusters. The presentation of these outputs is almost inevitably some variation of a word cloud. While it makes sense to present topics using words, we argue that word clouds only give a sense of concepts, not more developed thoughts or opinions. Longer, connected phrases and sentences provide people with more complete ideas, thoughts, and sentiments of the document authors.

A number of attempts to add more context or structure to word-based visualizations exist. One general approach is the use of semantic-preserving word clouds [11,31,38,44]. Another is to connect words through visual structures such as lines [9,22,26]. Yet another approach, including systems such as Word Tree [39], Phrase Net [35], and Wordgraph [32], positions and links words spatially following their natural occurrences in sentences, thus better encoding thoughts and opinions. These projects inspire our design of a novel visualization technique for summarizing text.

We introduce the **SentenTree**, a novel technique for visualizing the content of unstructured social media text. SentenTree seeks a balance between showing the most frequent words and preserving sentence structure. SentenTree gives people a high-level overview of the most common expressions in a document collection, and allows drilling down to details through interactions.

2 Related Work

Visualization has been applied to text documents of varying sizes for many different tasks [25]. Some projects facilitate the understanding of very large document sets by focusing on extracting the key themes and concepts and clustering documents according to these themes and concepts. Visualizations often map concepts and documents to a 2D/2.5D space and utilize spatial proximity to imply relationships between concepts and documents. Examples of this type of visualizations include

- *Mengdie Hu is with Georgia Institute of Technology. This work was conducted while she was interning with Twitter, Inc.. E-mail: mengdie.hu@gatech.edu.*
- *Krist Wongsuphasawat is with Twitter, Inc.. E-mail: kristw@twitter.com.*
- *John Stasko is with Georgia Institute of Technology. E-mail: stasko@cc.gatech.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

SPIRE/IN-SPIRE [41], the Stanford Dissertation Browser [7], and many knowledge mapping visualizations [3]. Complex clustering and language modeling techniques are often used to extract concepts and relationships, and the goal of these types of visualizations is understanding the landscape of documents beyond just learning about the content of the documents.

Another type of text visualization facilitates understanding of documents and insight generation by linking named entities. Jigsaw [16] is a system for investigative analysis that focuses on entity co-occurrence within the same document, while FacetAtlas [5] looks for complex multifaceted entity relationships.

Many text visualizations attempt to portray temporal changes of topics. The STREAMIT system [1] shows an evolving document collection as small circles that are clustered together to denote similarity. A popular visual metaphor used for portraying tempral changes is the streamgraph, as exemplified by ThemeRiver [17] that uses this technique to encode topic volume changes over time. MemeTracker [27], TIARA [40], TextFlow [10], and OpinionFlow [43] are just a few of the many visualizations using some variation of the ThemeRiver technique. Some of these systems overlay a word cloud on the ThemeRiver to display textual content. EventRiver [29], Leadline [14], Cloudlines [23], and Xu's system [45] use bubble-like visual elements to show event bursts detected from text streams. Another technique, employed by Story Tracker [24], clusters news stories and displays a set of keywords for each news story at a different time point. The keywords are linked over time by edges.

Other text visualizations map topics/concepts to physical locations. Scatterblogs [4] is an example system that generates keywords from social media text and overlays them on a map to show the origins of the text. Another system [28] combines a map view with word cloud and time series views to show how specific events have framed topic changes in document collections.

All of these visualizations seek to highlight concept relationships, topical changes, or physical locations in the visualization, and the textual content is either hidden or displayed primarily by a word cloud variation. Visualizations of the actual **textual content** are less common. In the next two sections we review two major classes of visualizations of textual content: word cloud-related visualizations and text structure visualizations.

2.1 Word cloud related visualizations

Tag clouds or word clouds are arguably the most widely used visualization method to display the content of text documents. Popularized by numerous websites, the original tag clouds visualizations show frequently used tags and vary font size according to tag usage frequency [36]. As people started to apply the visualization technique to other text documents, the name "word cloud" and sometimes "text cloud" grew in popularity. We will use "word cloud" to refer to this visualization technique hereafter.

A major variation on word clouds, the Wordle technique [37], automatically generates compact and aesthetically pleasing layouts with words at varying orientations. Chuang et al. [6] applied natural language processing techniques to improve word selection. Word clouds are also often added to other visual metaphors, as we discussed above, to provide some context to the textual content.

Despite their popularity, word clouds have been found to be lacking for analytical tasks [36]. Hearst and Rosner [18] pointed out three major problems with word clouds: 1) the size encoding is not accurate due to different word length, which makes it hard to compare words, 2) the physical layout is not meaningful, therefore words appear in discrete form and there's no context next to them, and 3) there is no natural "flow" for reading, the viewer just looks at random words in the visualization. Supporting the final two points, Yatani et al. found that humans tend to verbalize opinions in short expressions rather than discrete words [46].

One collection of projects, semantic-preserving word clouds, primarily address the second problem above by positioning words/terms closely to other related ones. Context-preserving word clouds [11], Seam-carving word clouds [44], ProjCloud [31], and ReCloud [38] all pursue this approach in different ways. In general, they focus on clustering related words together and providing compact visual representations of the word collections. Barth et al. [2] introduce three additional semantic word cloud algorithms and compare them to some of the other existing algorithms along various quantitative metrics. WordWanderer [13] allows a viewer to select or compare words in a cloud and then the position and appearance of other words changes to reflect the selection. All these techniques position words based on some measure of similarity or relatedness. SentenTree, conversely, uses actual sentence structures from the document collection to position words and terms, thus making it more like the systems discussed in the next section than the above techniques.

Another collection of projects extends word clouds by introducing graphical structures onto the word collections. Parallel Tag Clouds [9] shows connections between words in multiple tag clouds using a parallel coordinates style visualization. SparkClouds [26] does something similar, but uses sparklines as the organizing visual metaphor. WordBridge [22] shows clusters of words connected by lines and other "bridging" words that connect terms in different clusters. Our SentenTree technique similarly seeks to position and connect words using link structures, but they are derived from word positions in sentences.

2.2 Visualizations of text structures

Researchers have suggested other visualization techniques for textual content that preserve some of the original text and sentence structures. A notable example is the Word Tree [39], which allows a person to select a word of interest and displays the sentence segments next to that word by building and visualizing a prefix tree with the selected word at the root. The viewer thus learns about the word in the context of sentences. Double Tree [12] extends this technique by building Word Trees on both sides of the word of interest. Wordgraph [32] allows query by wildcard and displays not only Word Trees on both sides of the query pattern but also branches in between query terms. These techniques were a major source of inspiration to the design of SentenTree because we also seek to show words in the context of sentences. However, we do not want to require a person to select specific words to act as the foci of the visualization. Instead, we want the initial visualization to appear simply given a collection of documents.

The PhraseNet [35] visualization technique identifies word pair relationships (e.g., X and Y, X of Y, X's Y) in a document and displays those pairs in a node-link network. Although applied for a different purpose, showing uncertainty in statistically-derived lattice structures, the layout algorithm by Collins et al. [8] attempts to position words to reflect sentence structures with a goal similar to ours. It positions words from a sentence horizontally with potential replacement words drawn above others.

We also have been inspired by several infographics found online. One "genre" of these use flowchart-style graphs to show the sequential order and repetitions of words in items such as song lyrics and stories, as exemplified by a comic from xkcd [30].

3 SENTENTREE DESIGN

Figure 1 shows an example of the SentenTree visualization for a Twitter dataset of 189,450 tweets commenting on the opening game of the 2014 Soccer World Cup, posted during a 15 minute time window around the first goal. A person looking at this visualization will immediately notice some prominent words like *first*, *goal*, *world cup*, *watching*, etc. Similar to a text cloud, the large font size of these words indicates their high frequency of occurrence in the dataset. These words are good indications that people were discussing the game between Brazil and Croatia. An edge between two words indicates their occurrence in the same tweet. By interacting with the visualization and following the linked words from left to right, the viewer can further identify patterns such as *first goal world cup own goal*. Hovering the cursor over the word *own* (Figure 2a), highlights the expression *first goal world cup own goal*, fades other words, and displays example tweets containing these words in that order. Hovering over the word *score* (Figure 2b), highlights another expression: *brazil marcelo score first goal world cup brazil*. These actions should help the viewer learn that the first goal of the World Cup was an accidental own goal by Brazilian player Marcelo against his own team.

3.1 Design Goals

In this section we discuss four goals that drove the design of SentenTree. The first design goal was to leverage the positive qualities of word clouds, namely their ability to facilitate fast impressions by utilizing size. Since the most frequent words are encoded in the largest font sizes, they jump out to the viewer. In the SentenTree visualization, we also wanted the frequent words and expressions to be displayed more prominently so they can be easily spotted by the viewer.

The second goal was to bring in more sentence structure from the items in the text collection. As researchers have repeatedly found, a bag of discrete words is limited in its expressiveness. Other approaches, such as the Word Tree [39], give context to a word through displaying sentence fragments next to that word. Review Spotlight [46] uses adjective-noun pairs to summarize opinions, which turned out to be much more informative than text clouds for the same data. For SentenTree, we use fragments extracted from sentences to represent those sentences. For example, a person should be able to reasonably guess the meaning of *first goal world cup own goal* without reading the full sentence *the first goal of the World Cup is an own goal*. These fragments are called frequent sequential patterns, and we will more formally define them in the next section.

We chose to use patterns instead of full sentences because of a third design goal: the visualization should be concise but yet cover as much of the dataset as possible. A concise pattern may summarize many sentences that are similar to each other but have slight differences. For example, in the previous example, the pattern *first goal world cup own goal* is shared by 14,935 tweets (Figure 2a) and the pattern *score first goal world cup* is shared by 13,330 tweets (Figure 2b). Frequent sequential patterns are especially well suited to social media text because social media text collections on a given topic typically contain many sentences that share similar structures with small wording differences. The conciseness goal also suggests that we cannot show all frequent sequential patterns. Thus, the SentenTree technique collapses common parts of patterns to both save space and highlight their commonality. A reader familiar with the Word Tree might have noticed the similarity between Figure 1 and a Word Tree in that some large words have several branches extending from them on one or both sides. This is an indication that these big words are shared by several different sequential patterns.

Since our objective is to provide a high-level overview of the textual content, SentenTree follows the Shneiderman Mantra to provide an "overview first", then allow "zoom and filter" to get to "details on demand" [34]. By starting with the most common patterns, SentenTree addresses the entry point problem of the Word Tree [39]: instead of relying on people to identify their own entry point, the technique provides them with an overview of the most frequent patterns and allows them to select patterns of interest and drill down to see more details. This implies a fourth design goal on the technique: the pattern generation algorithm needs to be incremental.

3.2 Frequent Sequential Patterns

As discussed in the previous section, the central idea behind SentenTree is to take a large social media dataset, find the most frequent sequences of words, and build a visualization out from them. The sequences are called *frequent sequential patterns*. Below, we define the concept and then describe the sequence generation algorithm in detail. Because of the complexity of the algorithm, the following section provides an example to help explain its operations.

First, we formally define frequent sequential patterns for social media text based on a more general concept from data mining [33]. We consider a sentence a sequence of words. Suppose we have two sequences $\alpha =< a_1 a_2 \dots a_n >$, $\beta =< b_1 b_2 \dots b_m >$ where $a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m$ are words. We say that α is a subsequence of β, and β is a super-sequence of α, if there exist integers $1 \le j_1 < j_2 < \dots < j_n \le m$ such that $a_1 = b_{j_1}, a_2 = b_{j_2}, \dots, a_n = b_{j_n}$. This is denoted as $\alpha \subseteq \beta$.

The set of sentences containing the sequence α form the **support database** D_α. The **support** of α is the number of sentences in D. A sequence α is called a **frequent sequential pattern** when support of α exceeds some predefined lower threshold. In data mining, the threshold is called a **minimum support threshold** and is generally fixed for each mining task. In our case, the minimum support threshold is not fixed because we want to build frequent sequential patterns incrementally.

For example, suppose we have a database consisting of three sentences (sequences) s_1: *The first goal of the World Cup is an own goal*, s_2: *Someone somewhere has made a lot of money on first goal of World Cup being an own goal* and s_3: *Brazil's Marcelo scored the first goal of the World Cup*. The support of the sequence *first goal world cup* is 3 as it is a subsequence of all three sequences in the database, while the support of the sequence *first goal world cup own goal* is 2 because it is a subsequence of s_1 and s_2 but not s_3. If the minimum support threshold is 3, then only *first goal world cup* is considered a frequent sequential pattern. But if the minimum support threshold is 2, then both sequences are considered frequent sequential patterns.

```
Data: raw sentences
Result: graph
tokenized sentences = initialization(raw sentences);
create a pattern $s$ without any word and make $D_s$ = {all tokenized sentences};
list of leaf sequential patterns = patternGeneration($s$, $s$, default word count on screen);
construct graph out of leaf sequential patterns;
```

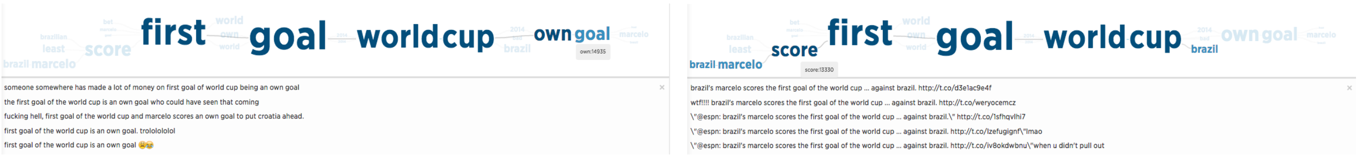
Algorithm 1: graphCreation()

```
Data: root node of tree of sequential patterns, start pattern,
      number of visible words needed
Result: list of leaf sequential patterns
if start pattern does not contain any word then
    number of visible words needed -= number of words in start pattern;
end
leaf sequential patterns = empty list;
push start pattern to leaf sequential patterns;
while leaf sequential patterns contains at least a word and visible word needed $> 0$ do
    $s$ = pop pattern with the largest support from leaf sequential patterns;
    if $s$ has no child sequences then
        find the most frequent super-sequence $s'$ of $s$ that is exactly one word longer than $s$;
        split $D_s$ into the support database for $s'$ and a new $D_s = D_s - D_{s'}$;
        add $s'$ as the left child of $s$, and the $s$ with new $D_s$ as right child of the old $s$;
    else
        $s'$ = the left child of $s$;
        $s$ = the right child of $s$;
    end
    number of visible words needed -= 1;
    push $s'$ and $s$ to leaf sequential patterns;
end
return leaf sequential patterns;
```

Algorithm 2: patternGeneration()

3.2.1 Sequential pattern generation and graph building algorithm

Algorithm 1 shows the process for constructing the graph data structure for the first time. The system first takes in raw sentences and goes through an initialization process (*initialization*()). This process normalizes sentences to lower case, performs tokenization to segment



someone somewhere has made a lot of money on first goal of world cup being an own goal
the first goal of the world cup is an own goal who could have seen that coming
fucking hell, first goal of the world cup and marcelo scores an own goal to put croatia ahead.
first goal of the world cup is an own goal. trololololol
first goal of the world cup is an own goal 😩😂

(a) Hovering over *own* in the World Cup visualization.

brazil's marcelo scores the first goal of the world cup ... against brazil. http://t.co/d3e1ac9e4f
wtf!!!! brazil's marcelo scores the first goal of the world cup ... against brazil. http://t.co/weryocemcz
\"@espn: brazil's marcelo scores the first goal of the world cup ... against brazil.\" http://t.co/1sfhqvlhi7
\"@espn: brazil's marcelo scores the first goal of the world cup ... against brazil. http://t.co/lzefugignf\"lmao
\"@espn: brazil's marcelo scores the first goal of the world cup ... against brazil. http://t.co/iv8okdwbnu\"when u didn't pull out

(b) Hovering over *score* in the World Cup visualization.

Fig. 2: Interacting with the World Cup visualization.

sentences into words (including numbers, hashtags, urls, etc.), and filters out stop words. Then an initial pattern is created without any words, and all tokenized sentences are put into its support database. This initial pattern serves as the root node of a tree of sequential patterns, which we use to store intermediate states of the sequential pattern generation process. This tree is important because we can reuse these intermediate states when zooming in and out on the visualization. Then the algorithm calls the *patternGeneration*() function to grow new sequential patterns from the root one.

The task of *patternGeneration*() (Algorithm 2) is to start with a given sequential pattern and grow its super-sequences until the total number of words in the patterns reaches the given number of visible words needed. These patterns will appear in the visualization and the given number of visible words needed is determined by the screen size (typically 100-200). The algorithm grows patterns by maintaining a list of leaf sequential patterns. At first, the only item in the list is the given sequential pattern. In every run, the most frequent pattern is popped from the leaf pattern list, and the program finds its most frequent super-sequence which is one word longer than the old pattern. This means a new word will be added to the visualization and the number of visible words needed is reduced by one. The new sequential pattern becomes the left child of the old sequential pattern on the tree, and a pattern that looks exactly like the old one is added as the right child of the old sequential pattern. The support database of the old pattern is split between the new patterns, and the new patterns are added to the list of leaf patterns. Therefore, at anytime, the original database is split between the support databases of all the leaf patterns. The program continues growing new leaf patterns until the number of visible words needed is reduced to zero.

Function *patternGeneration*() returns the list of leaf patterns to function *graphCreation*(), which uses these leaf patterns to construct a graph data structure for the visualization. In the graph data structure, each node is a word in the sequences; a word shared by multiple leaf patterns appears as one node. A directed edge is added between every pair of adjacent words in a leaf pattern.

After the visualization is created, the user may wish to zoom in on a frequent sequential pattern and bring up more children patterns. In this case, the program calls *patternGeneration*() with the selected sequential pattern as the start pattern and starts growing new patterns from there. The algorithm is usually able to reuse some patterns from the sequential patterns tree so it does not have to recalculate patterns that were discovered before. When the leaf patterns are returned, the program constructs a new graph from the leaf sequential patterns in the same way described in the previous paragraph.

Note the goal of this algorithm is different from a typical sequential pattern mining algorithm. Instead of trying to find **all** frequent sequential patterns based on a minimum support threshold, this algorithm only grows sequential patterns from existing patterns for building the graph visualization. Most algorithms in sequential pattern mining follow a depth-first approach to find sequential patterns because they want to enumerate all patterns satisfying a lower limit frequency. Our approach is breadth-first, and will take more time to perform than the depth-first approach if our goal is to exhaust all possible patterns. However, we only need a limited number of patterns each time due to display limits, therefore the breadth-first approach works well for its incremental quality.

3.2.2 Example - World Cup First Goal

We use a simplified World Cup example to illustrate our approach. Figure 3 illustrates the following steps.

We start with a dataset of 189,450 tweets. We first normalize the tweets to lower case, perform tokenization to segment each tweet into discrete words (including hashtags, urls, etc.) and remove stop words and punctuation.

1. We find the most frequent single word pattern (i.e. the most frequent word) in the dataset to be *goal*, and divide the dataset into tweets containing *goal* (74,554) and tweets without *goal* (114,896).

   leaf patterns: *empty pattern* (114,896), *goal* (74,554)

2. Next we pick the most frequent leaf pattern which is the empty pattern with 114,896 tweets. We find the most frequent single word pattern within these tweets to be *watching* (11,248) and add it to the tree, we also end up with an empty pattern with a support of 103,648.

   leaf patterns: *empty pattern* (103,648), *goal* (74,554), *watching* (11,248)

3. The most frequent leaf pattern is still the empty pattern. For illustration purpose we will ignore it from now on and pick the pattern *goal*. We will grow our pattern by one by finding the next most frequent pattern in the subset of tweets containing *goal*. The new sequential pattern is *first goal* (41,344). Note that the new pattern will always contain the previous pattern, as all tweets in this subset contain the previous pattern. We also end up with a pattern *goal* without a *first* before it and this subset has 33,210 tweets.

   leaf patterns: *first goal* (41,344), *goal* (33,210), *watching* (11,248)

4. The most frequent pattern is *first goal*. Create *first goal world*.

   leaf patterns: *first goal world* (36,136), *goal* (33,210), *watching* (11,248), *first goal* (5,208)

5. The most frequent pattern is *first goal world*. Create *first goal world cup*.

   leaf patterns: *first goal world cup* (36,081), *goal* (33,210), *watching* (11,248), *first goal* (5,208), *first goal world* (55)

6. The most frequent pattern is *first goal world cup*. Create *first goal world cup own*.

   leaf patterns: *goal* (33,210), *first goal world up* (21,220), *first goal world cup own* (14,861), *watching* (11,248), *first goal* (5,208), *first goal world* (55)

7. The most frequent pattern is *goal* (33,210). Create *own goal*. Note that in this new pattern *own goal* we have a word *own* and in a previously generated pattern *first goal world cup own* we also have a word *own*. But these two words are not considered common branches in the SentenTree and will be represented by two distinct nodes in the final visualization.

leaf patterns: *first goal world cup* (21,220), *goal* (19,977), *first goal world cup own* (14,861), *own goal* (13,233), *watching* (11,248), *first goal* (5,208), *first goal world* (55)

8. The most frequent pattern is *first goal world cup*. Create *score first goal world cup*. Note that each time we grow a new pattern, a word is added to the existing pattern. The new word can appear before, behind or in-between the words of the parent pattern.

   leaf patterns: *goal* (19,977), *first goal world cup own* (14,861), *score first goal world cup* (13,291), *own goal* (13,233), *watching* (11,248), *first goal world cup* (7,929), *first goal* (5,208), *first goal world* (55)

The leaf patterns *first goal world cup own*, *score first goal world cup*, *own goal*, and *watching* are used to construct a graph structure for the final visualization (Figure 4). Note that *first goal world cup own* and *score first goal world cup* share a subsequence *first goal world cup*, and this subsequence also shares the word *goal* with *own goal*. While *watching* is not connected to the other words and forms its own graph. We will discuss the layout and visual encodings of the graph in the next section. The different sequential patterns can be highlighted by hovering the mouse over one of the words. We will discuss the interactions in detail in section 3.4.

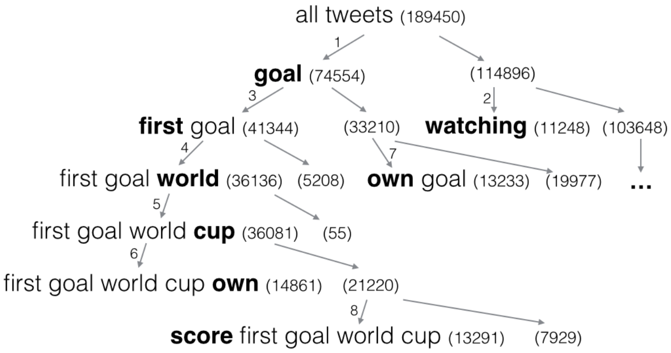


Fig. 3: An example pattern generation process. The words in boldface are new words added to the parent pattern to generate the current pattern. The numbers in parenthesis are the support of each sequential pattern.

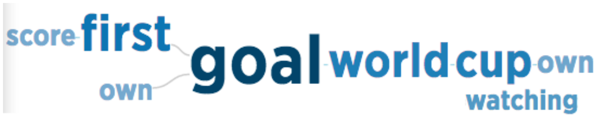


Fig. 4: A simple SentenTree of top sequential patterns in the World Cup dataset.

3.2.3 Additional Considerations

"Big words"

As described in previous sections, SentenTree prioritizes the most frequent sequential patterns and grows new patterns out of the existing ones. This introduces a problem in some scenarios: a large pattern may dominate the view and prevent other interesting patterns from surfacing. This is especially likely to happen when the dataset is retrieved based on particular keywords, therefore those keywords exist in most entries in the dataset. Since the person using the system is already familiar with the keywords, we suspect they are not crucial to the view. Additionally, their large size and visual dominance may obscure useful new information. Figure 5 shows what happens with the World Cup dataset used in the teaser image (Figure 1) without deprioritizing *world cup*. Note how *world cup* is huge and makes many branches small and difficult to read. Only one graph is in the view, so it generates significant white space, compared to Figure 1, where multiple graphs fill up the screen.

We address this problem by imposing a rule that words that appear in a large of number entries cannot be used to generate the first pattern of a graph. After testing a variety of values, we set the cut-off to be one third of the database size. This rule ensures that the view never ends up with a huge pattern that is significantly larger than the rest of its branches. These patterns will still show up in the visualization, but they will appear in multiple graphs so each one is smaller and does not skew the other words.

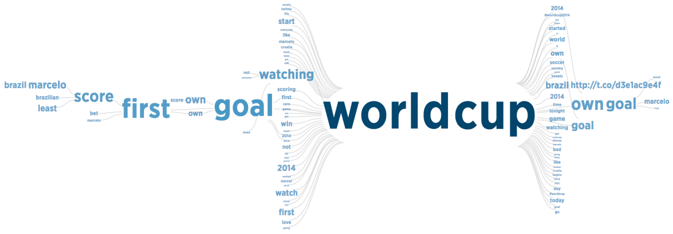


Fig. 5: A SentenTree visualization of the same dataset as Figure 1 without deprioritizing *world cup*.

Tokenization and filtering

Tokenization segments sentences into words. Our implementation employs a regular expression pattern to match not only words but also numbers, hashtags, urls, @ handles, etc., because they are prevalent in social media text. The current implementation of SentenTree does not perform stemming; therefore, words like *watch* and *watching* are considered different tokens. We are considering including a stemmer for future versions of SentenTree.

The SentenTree algorithm does not include punctuation as tokens. Additionally, it discards stop words such as "the", "and", "of", etc [1]. These choices are opposite as done in the Word Tree, which keeps both stop words and punctuation to best preserve the context for each word. We decided to remove stop words and punctuation because SentenTree is a high-level summary of the text. Stop words and punctuation are not likely to be as informative as substantive words in the dataset. Furthermore, common stop words and punctuation may "wash out" the more important content words. Note that negation words like "not" are not considered stop words because taking them out would reverse the meaning of sentences.

Currently the algorithm does not filter out hashtags or urls, but we have noticed that in general they are less informative than regular words and may be a waste of screen real estate, so we plan to include options to filter them out in the future.

3.3 Spatial layout and visual encodings

We produce a visualization from the graph structure generated in the previous section. The graph visualization is created through a force-directed approach. Each graph is its own SVG element so that the layout algorithm can run in parallel for multiple disconnected graphs. The SVG elements are ordered by the frequency of the largest word in the graph and packed as tightly as possible on the screen.

We developed alignment constraints for the force-directed layout and enforce them using the CoLa package [15]. The most basic constraint is the **word order** constraint: if two words appear in the same sequential pattern, the relative horizontal placement of words must follow their natural order in the pattern. This promotes that a person can read a pattern from left to right and understand its meaning. By experimenting with initial layouts, we further developed vertical and horizontal constraints that increase the legibility of the graph:

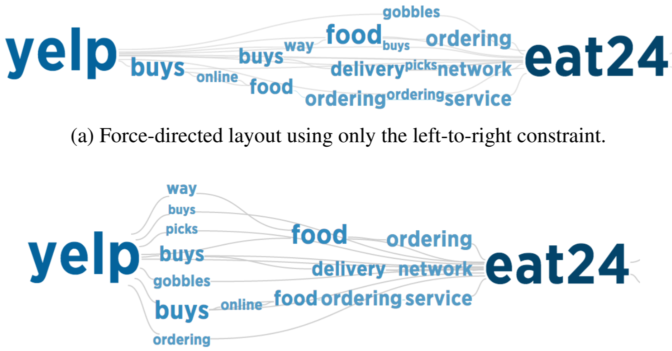
vertical: If two words always appear as a bigram, then we shorten the link between the two words and make sure they always appear on

[1] SentenTree uses a modified list based on the English stop words corpus from the Natural Language Toolkit (NLTK).

the same vertical level. An example is the words *world* and *cup* in Figure 1.

horizontal: If two patterns share some a common subpattern, then the words of the same distance from the common subpattern should center horizontally. An example is Figure 6b where the words *way*, *buys*, *picks*, etc. are centered horizontally because they all appear next to *yelp*.

These constraints not only make the graph layout less cluttered, but they also impose structure on the layout so that words in similar syntactic positions align vertically and can be compared against each other. The power of these layout constraints is demonstrated in Figure 6. This dataset consists of tweets mentioning the food ordering service Eat24 shortly after it was acquired by Yelp. (Only part of the visualization is shown in Figure 6.) Figure 6a shows the result of running the force-directed layout without the horizontal and vertical constraints, and Figure 6b shows the result with the constraints. Note that alignment yields useful information: the observer can see that when people discuss the acquisition, they use many sentences that are similar in form and meaning but vary slightly in wording. For example, they use *picks* and *gobbles* in place of *buys*, and describe Eat24 as *food ordering service* or *delivery network*.



(a) Force-directed layout using only the left-to-right constraint.

(b) Force-directed layout with horizontal and vertical constraints added.

Fig. 6: Part of a SentenTree visualizations of tweets discussing Yelp's acquisition of Eat24.

We use font size and color shading to double-encode the frequency of occurrence. We make the font size of a word proportional to the square root of the number of text documents containing the sequential pattern where the word first arises. Using the simplified World Cup example from the previous section, the size of *goal* is proportional to the square root of the number of tweets containing the pattern *goal*, while the size of *cup* is determined by the square root of the number of tweets containing the pattern *first goal world cup*. More frequent words are in a darker shade of blue than less frequent words. We also made it optional to turn the first and last word in a pattern to purple in order to distinguish the beginning and ending of patterns. Some of the use cases in this paper have that option turned on. We are considering other options for word color, such as encoding the sentiment of words, but the benefit provided by such a change must be weighed carefully against the visual variation and inconsistency it introduces. We also experimented with varying the width and shade of the edges but found that this introduces more visual clutter than useful information, so we decided to render the edges as thin light-gray curves.

3.4 Interactions

Since we are placing sequential patterns in a graph, a problem arises in that people viewing the visualization often cannot tell where a sequence starts and where it ends. As shown in Figure 4, someone viewing this visualization might assume a pattern *score first goal world cup own* exists, but in reality the dataset only contains *score first goal world cup* and *first goal world cup own* which are connected by the common part in-between. This problem is also present in Double Tree (Word Tree on both sides) [12] and Wordgraph [32].

We address this problem by using interaction. A person can hover the mouse over a word and all other words besides those that appear in its sequential pattern will become semi-transparent. The highlighted sequential pattern is the most frequent "leaf pattern" containing the selected word. (A "leaf" pattern is a pattern without a longer super-sequence on the screen.) Most words in the leaf pattern are colored in light blue but the words that appear as many or more times than the selected word are colored in dark blue. These words form the most frequent pattern containing the selected word, and a tooltip pops up showing the frequency of this pattern.

When a person hovers the mouse over a word, the system also displays the most common example sentences (e.g., Tweets) containing it in the lower left of the window, as shown in Figure 2. This helps the viewer learn more about precise thoughts and opinions in the text.

We also enable drilling down to an existing sequential pattern to see more details. When a person clicks on a word, SentenTree zooms in to the most frequent pattern containing the selected work (the pattern is colored in dark blue) and filters out all other sequential patterns. SentenTree also grows the current sequential pattern to include new words. An example is Figure 8a. The viewer clicks on *penalty* and all other branches disappear. The branch with *penalty* in the center grows out to fill the screen. The viewer can click on a RESET button in the interface (not shown in the figure) to go back to the full view.

3.5 Implementation and performance

We have implemented the SentenTree algorithm in Javascript for the web. In this implementation, a person provides raw text (e.g., tweets) to the application. All following computations are performed in the browser on the client side. We use d3.js[2] for the visualization and cola.js[3] for the constraint-based force-directed layout.

The pipeline of SentenTree can be broken into three steps: 1) load and preprocess (e.g. tokenize) the input data, 2) extract frequent sequential patterns and construct the graph data structure, 3) visualize the graphs on the screen. When a user interacts with the view by drilling down to show more details, we repeat steps 2 and 3, though the patterns generated from a prior step 2 can be reused.

The runtime of step 1 is linear to the total number of words in the dataset. Because the number of words in a sentence (or other social media text unit such as a Tweet) is limited, the runtime of step 1 is roughly linear to the number of sentences in the dataset. The runtime of step 2 is linear to the product of the number of sentences and the number of words in the final visualization. Because we limit the number of words shown due to available screen space, the runtime of step 2 is also roughly linear to the number of sentences in the dataset. The runtime of step 3 is more difficult to estimate, because the constraint-based force-directed layout is influenced both by the number of words and edges in the graph as well as the complexity of the graph. The layout algorithm runs in parallel for multiple disconnected graphs, so the time consumed is dependent on the largest, most complicated graph.

We tested the technique on datasets with 10,000 (10K), 100,000 (100K), and 1,000,000 (1M) unique sentences in a Google Chrome Browser on a MacBook Air laptop. The number of words shown in the visualization was fixed at 150. For the 10k datasets, step 1 took 0.25 to 0.45 seconds, and step 2 took 0.8 to 2 seconds. For the 100k datasets, step 1 took under 3.1 seconds, and step 2 took 11 to 17.5 seconds. For the 1M datasets, step 1 took under 30 seconds, and step 2 took under 1 minute. The runtime for step 3 is not related to the input data size, but is dependent on the most complicated graph in the visualization. For all of the datasets we tested, the most time-consuming graph layout took under 6 seconds, though we observed that layout started to stabilize after the first half second and only made minor movements afterwards. Thus, the effective total duration to display each of the three sizes of data was about 2 seconds, 20 seconds, and 2 minutes.

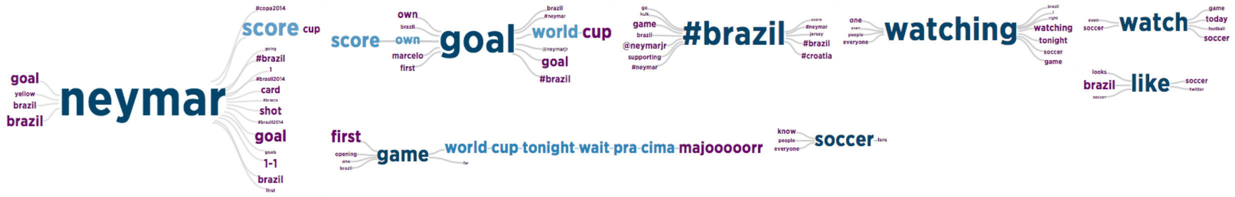
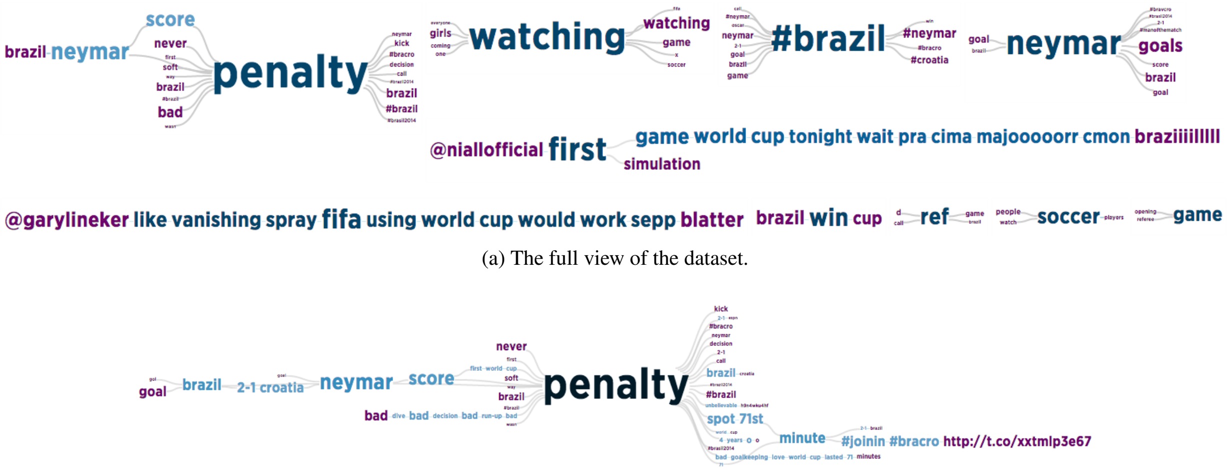


Fig. 7: A SentenTree visualization of tweets commenting on the second goal of the opening game of the World Cup. This dataset contains 135,841 tweets (81,253 unique tweets).



(a) The full view of the dataset.

(b) A zoomed-in view focused on *penalty* after the viewer has clicked on that word.

Fig. 8: A SentenTree visualization of tweets commenting on the third goal of the opening game of the World Cup. This dataset contains 132,599 tweets (75,930 unique tweets).

4 Example Use Cases

4.1 A typical exploration scenario - World Cup

In this section, we describe a use case of exploring comments on an event through SentenTree visualizations. Suppose a person wishes to learn about Twitter users' reactions to the opening game of the 2014 Soccer World Cup between Brazil and Croatia. The person already knows that multiple goals were scored during the game, each creating a huge spike in tweet volume. For each goal, the person is able to obtain 15 minutes of relevant tweets using keywords and hashtags like "world cup", "#worldcup2014", "#bravscro", "#bra", "#cro", etc. We describe how she explores the tweets for the first three goals of the game.

First Goal

The SentenTree visualization for the first goal is presented in Figure 1. As previously discussed, the person immediately notices a large pattern *first goal world cup*. Hovering her mouse over *cup* tells her that this pattern appears 36,081 times. In other words, close to 40% of tweets about the World Cup posted in this 15 minutes time window include the pattern *first goal world cup*. As she hovers the mouse over a few other branches, she notices patterns like *brazil marcelo score first goal world cup* and *first own goal world cup history*. After exploring more branches and reading some example tweets, she concludes that most Tweets were either describing what happened on the soccer field or expressing excitement over the goal and surprise at Marcelo's big blunder.

Second Goal

The person moves on to the second goal of the game (Figure 7). She immediately notices that the biggest branch is centered around *neymar*. Hovering over the branches she notices people mentioning that Brazilian player Neymar had scored the second goal of the game, which brings the score to 1-1. She also notices branches like *yellow Neymar* and *Neymar card*. She clicks on them to bring up example tweets explaining that Neymar had drawn a yellow card minutes before he scored the goal. The person clicks on *Neymar* to bring up a detailed view. This view directs her to longer patterns about Neymar which she explores for a while, learning that at age 22 Neymar was considered a "boy wonder" and this was the 32nd goal he scored for Brazil, making him the third highest Brazilian goalscorer.

Zooming back to the full view, she also notices a branch centered around *goal*. Hover her mouse over the branches, she finds patterns such as *own goal*, *score own goal*, and *marcelo goal* which indicates that people were still discussing the first goal of the game being an own goal. She also saw a big branch under *game* which reads *first game world cup tonight wait par coma majoooorr*. Intrigued, she clicks on the branch to bring up an example tweet and discovers that people were retweeting an earlier tweet by Niall Horan which says "First game of the World Cup tonight! Can't wait! PRA CIMA MAJOOOOORR! CMON BRAZIIIILLLLL!". Niall Horan is a member of the popular boy band One Direction and one of the most followed celebrities on Twitter. Therefore it is not surprising that his tweet was retweeted so many times.

Third Goal

The person moves on to the third goal of the game (Figure 8a). She sees that a large branch is centered around the word *penalty* and another branch is centered around *neymar*. Hovering the mouse over a few branches she learns that Neymar scored a penalty kick for Brazil making the score 2-1. She is interested in learning more about people's reaction to the penalty goal, so she zooms in on *penalty*. Figure 8b shows a detailed view with *penalty* at the root. The person notices some new words connected to *penalty*, such as *bad*, *soft*, *never* to its left and *unbelievable* and *bad* to its right. She hovers the mouse over these words to see the branches and also brings up a few example tweets. She learns that the penalty decision was controversial, as Twitter users call it "wasn't a penalty" ,"never a penalty", "soft penalty", "bad penalty",

[2] http://d3js.org/

[3] http://marvl.infotech.monash.edu/webcola/

"worst penalty decision", and even "unbelievable".

Zooming out and looking at the full view, the person notices that Niall Horan's tweet was still being retweeted by many followers. She also notices a tweet by @garylineker with many retweets. Clicking on the branch she brings up the original tweet. It appears to be a rather unflattering joke towards FIFA president Sepp Blatter which was echoed by over 3,000 Twitter users. (@GaryLineker: "I like this vanishing spray FIFA are using for the World Cup. Would it work on Sepp Blatter?")

By going through the three goal visualizations, the person was able to form a coherent narrative of not only what happened on the field, but also how people reacted to each goal. She discovered that people were shocked by the first own goal, applauded the second goal by Neymar, and were unhappy with the penalty goal which gave Brazil a lead in the game. They were excited by the dramatic opening game and expectant of the rest of the World Cup.

This is a typical use case of the SentenTree visualization for exploring a large dataset (or datasets). In the initial stage, SentenTree supports impression-forming just as a word cloud. In addition, it provides context for each word by placing the words within sequential patterns and allows easy highlighting of patterns through hovering. When interested in a sequential pattern the viewer can click on it to drill down to see more details. The viewer also can click on sequential patterns to bring up example sentences. Because Tweet-reading typically happens after the viewer has formed initial expressions of the dataset, the person only needs to bring up sentences that look relevant instead of drowning in a sea of text.

4.2 Natural content clustering

In the previous section, we illustrated the use of the SentenTree visualization to explore social media reactions to a high-profile event. Based on our experiences applying SentenTree to different datasets, we have observed that for datasets with somewhat diverse content, SentenTree often results in natural clustering of texts. In this section, we give three examples of natural clustering and discuss how SentenTree can be used to explore these datasets.

4.2.1 Communicating multiple word meanings

We obtained a dataset of tweets posted on Aug 1, 2014 with the keyword "yosemite". As *yosemite* is present in every tweet in the dataset, it is not allowed in the first step of sequential pattern generation. The resulting visualization is shown in Figure 9. Yosemite is a word with ambiguous usage, and this is reflected in the visualization. We observe that the biggest branch is about OS X Yosemite (a version of the Mac operation system). Another smaller branch is also about the Mac OS with *Apple* as the most prominent word in the branch. Another usage of the word "yosemite" is the name of a national park in California. We find one branch with *yosemite national park california* and another branch about a wildfire in the park. SentenTree is able to separate tweets on the Mac OS and the national park without any semantic analysis.

4.2.2 Communicating multiple concepts

A SentenTree visualization of tweets about the startup Eat24 retrieved shortly after Yelp's announcement to acquire the company resulted in three clusters. (Please refer to the supplement material for the image.) The first is a branch focused around the sequential pattern *yelp eat24* about the news of the acquisition worded in slightly different ways. The second is a branch with *@eat24* at the center that appears to be Eat24's official Twitter account which interacts with its customers frequently. From the visualization we learn that customers communicate with @eat24 to talk about the mobile app, coupons, their order, etc. A final small branch has *eat24* without *yelp*. Expanding the branch, we discover that tweets in this branch are focused on Eat24's Super Bowl commercial with rapper Snoop Dogg.

4.2.3 Communicating multiple facets

So far, all examples we have shown are tweets. We include a final dataset of consumer reviews to illustrate another application of the technique. We include this case to demonstrate SentenTree's power to highlight different facets in a dataset. The dataset includes reviews crawled from Amazon.com about a Samsung TV model. As the unit of analysis is a single sentence, we segment the reviews into sentences before feeding them to SentenTree. We also specify a rule that *samsung* and *tv* should not be used in the first round when generating sequential patterns as we do not want these words to dominate the view.

Figure 10 is the SentenTree visualization of the dataset. The visualization consists of several branches each having a bigger word in the center and a set of smaller words on both sides. These branches are reminiscent of a double-sided Word Tree (e.g. the Double Tree). This indicates that the data set does not have longer frequent sequential patterns. The likely reasons are 1) the dataset is considerably smaller than the previous tweet datasets and there simply are not enough sentences to form similar long sentences, and 2) in consumer reviews people tend to write longer paragraphs with more variations in their expressions than what people tend to write on Twitter.

Although there are no long patterns, the branches give indications to different facets people frequently mention. There are *picture*, *sound*, *amazon*, *lcd* which are all product/services facets of the TV. By browsing the words used together with each facet word, the viewer is able to learn what people like and/or dislike about each facet. Major branches with *great* and with *good* at the center are evident. By looking at these branches the viewer is able to figure out that reviewers report positive feelings about the picture, the price, Amazon, and the TV in general.

Consumer reviews research suggests representing facets as nouns and using adjectives or phrases close to the nouns to describe each facet [20, 21, 46]. Although SentenTree does not perform parts-of-speech analysis, facets (such as "picture", "sound", etc.) naturally emerge because the words are used frequently by reviewers. Additionally the words that frequently co-occur with facet words are shown on either side of the facet words. We argue that SentenTree serves well as a generic tool for someone casually exploring a review dataset, especially considering it does not perform complex natural language processing.

5 Initial User Feedback

We showed SentenTree to three people to gain initial feedback about the system. All three are research scientists or data scientists at technology companies who regularly work with social media text. We gave each a short demo of SentenTree and asked them to use the system to analyze a few Twitter datasets. We observed how they interacted with the visualization and asked for feedback at the end of the session.

All three people immediately understood the correspondence between font size/color and frequency, and all of them spent significant time hovering the cursor over different words in the view to read example tweets. The participants had to be reminded of the less discoverable functionalities (such as clicking to drill into a pattern, searching, and zooming). It also initially surprised them when they saw the same word multiple times in the visualization, but once they were given an explanation and used hover to check that the word appeared in different patterns, they became comfortable with the concept.

All three people found SentenTree enjoyable to use and commented that it added fun to reading text. They also found SentenTree effective for helping to explore text datasets.

The aspects of the technique receiving most positive comments include:

- The most frequent words and patterns "pop out" in the visualization because they are larger and darker, and the frequency count can be read in a tooltip.
- The context to a word or pattern is immediately available by cursor hover.
- Similar content is grouped into a graph or branch that makes it easy to form an impression of the major topics.
- The visualization can serve as a filtering tool and greatly saves time in finding interesting content to read.
- Less frequent content is not eliminated; instead it is put into small but discoverable branches.
- One can keep drilling down into interesting branches and search for words on the screen.

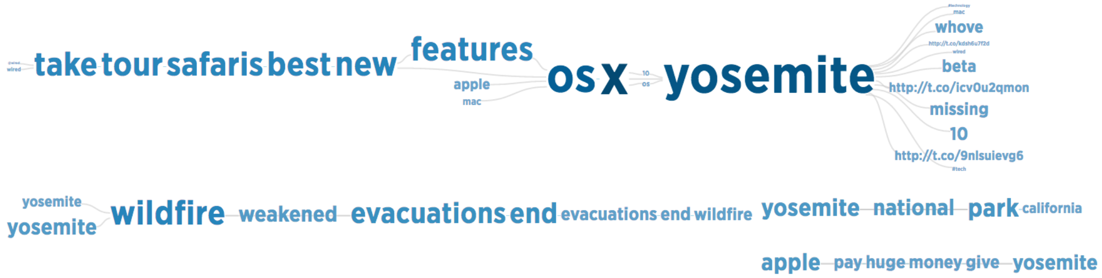


Fig. 9: A partial SentenTree visualization of an entire day's (August 1st, 2014) tweets on the subject "yosemite". The dataset contains 6,712 tweets (4,963 unique tweets).

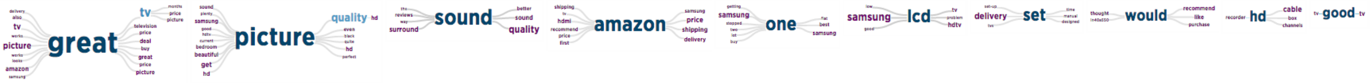


Fig. 10: A SentenTree visualization of Amazon.com Reviews on a Samsung TV model. The dataset contains 109 reviews with 941 sentences in total. The unit of analysis is a sentence.

The participants also pointed out a few items to improve on or to add to SentenTree:

- A common request was to perform stemming and spell checks, and to merge different forms of the same word into a single item.
- Improvements to the UI such as packing the graphs tightly to reduce white space and sorting vertically parallel branches by frequency or alphabetic order.
- Perform sentiment analysis and color-code words by sentiment.
- Render the example text in its original form. Some of the tweets they saw contained images that provide important context and it may be helpful to render them inline.
- Reduce the noise in the visualization, e.g., provide an option to filter out certain words such as hashtags.
- Provide the ability to filter on meta-information. For example, one of the people works with geographic information regularly and asked for the ability to filter text based on author location.

6 Discussion

We designed the SentenTree technique to provide a number of the key benefits evident in both word clouds and the Word Tree, while also overcoming some of the limitations of each. SentenTree highlights the important (most frequent) words as do word clouds, but it fills in the connections between words to communicate sentence structure and underlying concepts, themes, and ideas more fully. A Word Tree shows the most frequent prior and following words and sentence fragments clearly, but a person must select a word to serve as the focus for the visualization. SentenTree, conversely, constructs a network of sentence fragments and automatically extracts and shows different words that effectively serve as foci within the visualization.

Additionally, the interactive capabilities of SentenTree allow a person to see specific connected sets of words just as they would appear, in order, in social media posts. A person can use interaction to effectively "drill-down" and display the individual postings from which words are taken. We envision SentenTree as a helpful component in many other text and document visualization systems. SentenTree is designed in a modular way so it may be customized or integrated into a multi-view system. Multiple people to whom we have demoed the system have commented that they would like to incorporate SentenTree as a view in their systems.

Like many research techniques, SentenTree has limitations. It is arguably less intuitive to interpret than word clouds and the Word Tree in its raw static form. Viewers may infer patterns or sequences of words that do not occur within the collection unless they use the interactive capabilities of the system. That reliance on interaction, which in general provides power to a visualization, is also a potential shortcoming.

The technique also does not pack visual elements (i.e., words) in a dense and efficient manner into a rectangular region.[4] Often, visualizations produced with the system exhibit quite a bit of whitespace. For tools where screen real estate is at a premium, this can be a problem.

Another limitation of the current technique is that it does not explicitly communicate any information about the temporal ordering of the social media messages it is portraying. As we demonstrated with the World Cup dataset, each SentenTree visualization is a snapshot of text between a period of time, and the viewer has to create multiple visualizations to see how content changes through time.

Finally, as discussed in Section 3.5, the algorithm for generating a SentenTree does not run in interactive real time. For collections around 100,000 social media messages, the system takes about 20 seconds to produce a final visualization that is ready for viewing and interaction. Ideally, SentenTree could be integrated into a surrounding system in a way that would lessen the impact of waiting. However, the time to run the algorithm means the visualization is not dynamic and cannot be changed on the fly. For example, SentenTree does not support merging or removing of words in the view because removing a frequent word would cause all patterns containing that world to be re-created. In that case, the final graphs might look completely different.

7 Conclusion

We have presented SentenTree, a novel visualization technique for social media text. By visualizing frequent sequential patterns, we have sought to find a sweet spot between displaying discrete words and showing full sentences. We described an algorithm for incrementally generating these frequent sequential patterns and visualizing them. Through a number of examples, we also illustrated how SentenTree can help people gain a rapid understanding of key concepts and opinions in a large text collection.

For future work, we plan to adopt some of the useful feedback gathered from our initial user trials such as encoding sentiment, showing temporality, or providing dynamic controls. We also plan to evaluate the technique with more people, in particular, have people try the technique on data of interest to them. Eventually we plan to host SentenTree online and provide an interface for anyone to upload their own datasets. We want to learn how people explore their own data with SentenTree and test the application with very large datasets.

Acknowledgments

This research supported in part by the DARPA XDATA program and the National Science Foundation via award IIS-1320537.

[4] This fact made producing good figures for this article challenging.

REFERENCES

[1] J. Alsakran, Y. Chen, Y. Zhao, J. Yang, and D. Luo. Streamit: Dynamic visualization and interactive exploration of text streams. In *Proceedings of 2011 IEEE Pacific Visualization Symposium*, pages 131–138, Mar 2011.

[2] L. Barth, S. G. Kobourov, and S. Pupyrev. *Experimental Comparison of Semantic Word Clouds*, pages 247–258. Springer International Publishing, 2014.

[3] K. Börner. *Atlas of Science - Anyone can Map*. MIT Press, 2015.

[4] H. Bosch, D. Thom, M. Worner, S. Koch, E. Puttmann, D. Jackle, and T. Ertl. Scatterblogs: Geo-spatial document analysis. In *Proceedings of 2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 309–310, Oct 2011.

[5] N. Cao, J. Sun, Y.-R. Lin, D. Gotz, S. Liu, and H. Qu. Facetatlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1172–1181, 2010.

[6] J. Chuang, C. D. Manning, and J. Heer. Without the clutter of unimportant words: Descriptive keyphrases for text visualization. *ACM Transactions on Computer-Human Interaction*, 19(3):1–29, 2012.

[7] J. Chuang, D. Ramage, C. Manning, and J. Heer. Interpretation and Trust: Designing Model-driven Visualizations for Text Analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 443–452, May 2012.

[8] C. Collins, S. Carpendale, and G. Penn. Visualization of uncertainty in lattices to support decision-making. In *Proceedings of the 2007 Joint Eurographics / IEEE VGTC Conference on Visualization (EuroVis)*, pages 51–58, Jun 2007.

[9] C. Collins, F. B. Viégas, and M. Wattenberg. Parallel tag clouds to explore and analyze faceted text corpora. In *Proceedings of 2009 IEEE Symposium on Visual Analytics Science and Technology, (VAST)*, pages 91–98.

[10] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong. Textflow: Towards better understanding of evolving topics in text. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2412–2421, 2011.

[11] W. Cui, Y. Wu, S. Liu, F. Wei, M. X. Zhou, and H. Qu. Context-preserving, dynamic word cloud visualization. *IEEE Computer Graphics & Applications*, 30(6):42–53, 2010.

[12] C. Culy and V. Lyding. Double tree: an advanced kwic visualization for expert users. In *Proceedings of 2010 14th International Conference on Information Visualisation (IV)*, pages 98–103, Jul 2010.

[13] M. Dörk and D. Knight. WordWanderer: a navigational approach to text visualisation. *Corpora*, 10(1):83–94, 2015.

[14] W. Dou, X. Wang, D. Skau, W. Ribarsky, and M. X. Zhou. Leadline: Interactive visual analysis of text data through event identification and exploration. In *Proceedings of 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 93–102, Oct 2012.

[15] T. Dwyer, Y. Koren, and K. Marriott. IPSep-CoLa: An Incremental Procedure for Separation Constraint Layout of Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):821–828, 2006.

[16] C. Görg, Z. Liu, J. Kihm, J. Choo, H. Park, and J. Stasko. Combining computational analyses and interactive visualization for document exploration and sensemaking in Jigsaw. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1646–1663, 2013.

[17] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

[18] M. A. Hearst. What's Up with Tag Clouds? *Visual Business Intelligence Newsletter*, 2008.

[19] M. Hu, S. Liu, F. Wei, Y. Wu, J. Stasko, and K.-L. Ma. Breaking news on twitter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2751–2754, May 2012.

[20] M. Hu, H. Yang, M. X. Zhou, L. Gou, Y. Li, and E. Haber. OpinionBlocks: a crowd-powered, self-improving interactive visual analytic system for understanding opinion text. In *Proceedings of INTERACT 2013*, pages 116–134. Sep 2013.

[21] J. Huang, O. Etzioni, L. Zettlemoyer, K. Clark, and C. Lee. Revminer: An extractive interface for navigating reviews on a smartphone. In *Proceedings of the 2012 ACM UIST Symposium*, pages 3–12, Oct 2012.

[22] K. Kim, S. Ko, N. Elmqvist, and D. S. Ebert. Wordbridge: Using composite tag clouds in node-link diagrams for visualizing content and relations in text corpora. In *Proceedings of the 2011 Hawaii International Conference on System Sciences (HICSS)*, pages 1–8, Jan 2011.

[23] M. Krstajic, E. Bertini, and D. A. Keim. Cloudlines: Compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.

[24] M. Krstaji, M. Najm-Araghi, F. Mansmann, and D. A. Keim. Story Tracker: Incremental visual text analytics of news story development. *Information Visualization*, 12(3-4):308–323, 2013.

[25] K. Kucher and A. Kerren. Text visualization techniques: Taxonomy, visual survey, and community insights. In *Proceedings of 2015 IEEE Pacific Visualization Symposium*, pages 117–121, April 2015.

[26] B. Lee, N. H. Riche, A. K. Karlson, and S. Carpendale. Sparkclouds: Visualizing trends in tag clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1182–1189, 2010.

[27] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 497–506, Jun 2009.

[28] Y. Lu, M. Steptoe, S. Burke, H. Wang, J. Y. Tsai, H. Davulcu, D. Montgomery, S. R. Corman, and R. Maciejewski. Exploring evolving media discourse through event cueing. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):220–229, 2016.

[29] D. Luo, J. Yang, M. Krstajic, J. Fan, W. Ribarsky, and D. Keim. EventRiver: interactive visual exploration of constantly evolving text collections. *IEEE Transactions on Visualization and Computer Graphics*, 2010.

[30] R. Munroe. xkcd. `http://xkcd.com/851`. Accessed: June 19, 2016.

[31] F. V. Paulovich, F. M. B. Toledo, G. P. Telles, R. Minghim, and L. G. Nonato. Semantic wordification of document collections. *Computer Graphics Forum*, 31(3):1145–1153, 2012.

[32] P. Riehmann, H. Gruendl, M. Potthast, M. Trenkmann, B. Stein, and B. Froehlich. Wordgraph: Keyword-in-context visualization for netspeak's wildcard search. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1411–1423, 2012.

[33] W. Shen, J. Wang, and J. Han. Sequential Pattern Mining. In *Frequent Pattern Mining*, pages 261–282. Springer, 2014.

[34] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–343, Sep 1996.

[35] F. Van Ham, M. Wattenberg, and F. B. Viégas. Mapping text with phrase nets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1169–1176, 2009.

[36] F. B. Viégas and M. Wattenberg. Timelines tag clouds and the case for vernacular visualization. *interactions*, 15(4):49–52, 2008.

[37] F. B. Viégas, M. Wattenberg, and J. Feinberg. Participatory visualization with Wordle. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1137–1144, 2009.

[38] J. Wang, J. Zhao, S. Guo, C. North, and N. Ramakrishnan. ReCloud: Semantics-based word cloud visualization of user reviews. In *Proceedings of Graphics Interface 2014*, pages 151–158, May 2014.

[39] M. Wattenberg and F. B. Viégas. The word tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1221–1228, 2008.

[40] F. Wei, S. Liu, Y. Song, S. Pan, M. X. Zhou, W. Qian, L. Shi, L. Tan, and Q. Zhang. Tiara: a visual exploratory text analytic system. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 153–162, Jul 2010.

[41] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, pages 51–58, Oct 1995.

[42] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts. Who says what to whom on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, pages 705–714, Mar 2011.

[43] Y. Wu, S. Liu, K. Yan, M. Liu, and F. Wu. Opinionflow: Visual analysis of opinion diffusion on social media. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1763–1772, 2014.

[44] Y. Wu, T. Provan, F. Wei, S. Liu, and K.-L. Ma. Semantic-preservingword clouds by seam carving. *Computer Graphics Forum*, 30(3):741–750, 2011.

[45] P. Xu, Y. Wu, E. Wei, T.-Q. Peng, S. Liu, J. J. H. Zhu, and H. Qu. Visual analysis of topic competition on social media. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2012–2021, 2013.

[46] K. Yatani, M. Novati, A. Trusty, and K. N. Truong. Review spotlight: a user interface for summarizing user-generated reviews using adjective-noun word pairs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1541–1550, May 2011.